Types and Roles for Web Security

Types and Roles for Web Security

Silvia Ghilezan, Svetlana Jakšić, Jovanka Pantović and Mariangiola Dezani-Ciancaglini

***Abstract*—This paper summarises results obtained by the authors in studying security and privacy issues of web data. The proposed approach is based on typed process calculi that appear to be suitable for controlling access rights.**

***Index Terms*—types, security, XML, process calculi, role based access control.**

I. Introduction

As information networks become more open and dynamic, the need for protecting security and privacy of data is increasingly important in many fields of human activities. Systems must be able to exchange data and processes while preserving security. In case we are given a target security policy for a distributed system containing XML data, how can we check weather the system behaves according to the policy? One solution is to suitably annotate the security relevant events, to classify them according to a type system and to verify security properties by typing. Using a process calculus representation of a distributed network, in [1], [2] we assign security levels to the data and the processes of the network and prove relevant security properties. In [3] we extend our previous work with means for more efficient dynamic change of access rights. Below, we give an example which points out similarities and differences between our two approaches.

An example

Let us consider a simple distributed system consisting of four principals: an online music store, a visitor, a registered user and the owner of the store. Let the online store, written in XML notation, have the shape:

$$\begin{array}{l} <\texttt{store}> \\ \quad <\texttt{lyrics}> \\ \quad\quad title \\ \quad </\texttt{lyrics}> \\ \quad <\texttt{download}> \\ \quad\quad song \\ \quad </\texttt{download}> \\ </\texttt{store}> \end{array}$$

The store contains lyrics of a song under $title$ and the file $song$ for download. In order to describe the behaviour of the visitor, the registered user and the owner we use process calculus notation. So we will write

$$\texttt{read}_{\texttt{store/lyrics}}(\chi)$$

for a visitor wishing to read lyrics. The process

$$\texttt{read}_{\texttt{store/download}}(\chi)$$

represents a registered user wishing to download music and the process

$$\texttt{change}_{\texttt{store/download}}(\chi, demo).\texttt{enable}_{\texttt{store}}(\texttt{visitor})$$

represents the owner of the store replacing the $song$ file with a $demo$ file and then offering the whole contents of the store to any visitor for free. It is reasonable to ask that the store behaves according to the following security conditions:

1) a visitor of the store is allowed to read the lyrics,
2) a registered user is allowed to read the lyrics and download the music,
3) only the owner can decide to change access rights to the data in the store.

The first approach [1], [2] assigns security levels to data and processes, so the online store becomes:

$$\begin{array}{l} <\texttt{store}> \\ \quad <\texttt{lyrics}> \\ \quad\quad title^1 \\ \quad </\texttt{lyrics}> \\ \quad <\texttt{download}> \\ \quad\quad song^2 \\ \quad </\texttt{download}> \\ </\texttt{store}> \end{array}$$

More precisely in the calculus of [1], [2] the online store is represented by the following data tree

$$\texttt{store}\ [\texttt{lyrics}\ [title^1] \mid [\texttt{download}\ [song^2]] \qquad (1)$$

The security level of the lyrics $title$ is 1 and the security level of the $song$ file is 2. If we convene that an agent of level h can read data of level less than or equal to h and we assign level 1 to the visitor and level 2 to the user, then the first two security conditions will be satisfied. In this case the owner to show the demo to the visitor should replace the $song$ file of level 2 with a $demo$ file of level 1

$$\texttt{change}_{\texttt{store/download}}(\chi, demo^1) \qquad (2)$$

since this approach does not allow changes of access rights. The access control has to be modified by changing the data, so we can not say that the third security condition is satisfied. Therefore we have considered a new approach in which the emphasis is on dynamic changes of access right controls.

In order to accomplish that, in [3], we have introduced role-based access control into our model. Firstly let us take the

S. Ghilezan, S. Jakšić and J. Pantović are with Faculty of Technical Sciences, University of Novi Sad, Serbia, e-mail:{gsilvia,sjaksic,pantovic}@uns.ac.rs.

M. Dezani-Ciancaglini is with Dipartimento di Informatica, Università di Torino, Italy, e-mail:dezani@di.unito.it

online store to be:

$$\begin{array}{l} < \texttt{store role} = \texttt{visitor} > \\ \quad < \texttt{lyrics role} = \texttt{visitor} > \\ \quad\quad \mathit{title} \\ \quad < /\texttt{lyrics} > \\ \quad < \texttt{download role} = \texttt{user} > \\ \quad\quad \mathit{song} \\ \quad < /\texttt{download} > \\ < /\texttt{store} > \end{array}$$

or in the syntax of our calculus (using obvious abbreviations):

$$\texttt{store}^{\{\texttt{visitor}\}}[\texttt{ly}.^{\{\texttt{visitor}\}}[\mathit{title}]|[\texttt{down}.^{\{\texttt{user}\}}[\mathit{song}]] \quad (3)$$

In this way we have assigned a set of roles to each tag. Let the roles $\texttt{visitor}, \texttt{user}$ and $\texttt{owner}$ belong to a countable set of roles which is a lattice for the partial order $\sqsubseteq$. We consider the role $\texttt{visitor}$ to be lower than the role $\texttt{user}$ and the role $\texttt{user}$ to be lower than the role $\texttt{owner}$, i.e.: $\texttt{visitor} \sqsubseteq \texttt{user} \sqsubseteq \texttt{owner}$. As expected, we assign the role $\texttt{visitor}$ to the visitor, the role $\texttt{user}$ to the registered user and the role $\texttt{owner}$ to the owner of the online music store. We say that the tag (or the edge when we use tree representation of XML documents) $\texttt{store}$ is accessible to the process with role $\texttt{visitor}$ or higher. The path $\texttt{store}/\texttt{lyrics}$ is accessible to the process with the role $\texttt{visitor}$ since both tags are accessible to it, while the path $\texttt{store}/\texttt{download}$ is not. In this approach the locations have policies which regulate changes of access rights. For example if the store's location policy is

$$(\{\texttt{visitor}\}, \{(\{\texttt{owner}\}, \texttt{visitor})\}, \{(\{\texttt{owner}\}, \texttt{user})\}) \quad (4)$$

then the processes with a role lower then $\texttt{visitor}$ can not access the music store at all and that the owner may allow visitors or ban users to access the store. After the owner places demo file and allows all visitors to access, the store becomes:

$$\begin{array}{l} < \texttt{store role} = \texttt{visitor} > \\ \quad < \texttt{lyrics role} = \texttt{visitor} > \\ \quad\quad \mathit{title} \\ \quad < /\texttt{lyrics} > \\ \quad < \texttt{download role} = \{\texttt{visitor}, \texttt{user}\} > \\ \quad\quad \mathit{demo} \\ \quad < /\texttt{download} > \\ < /\texttt{store} > \end{array}$$

In this way also the last security condition is satisfied.

In the rest of this presentation we will relay on the given example and we will omit almost all technical details that can be found in the papers [1]–[3].

Paper Content

Sections II and III describe the syntax, types and security properties of the X$d\pi$ calculus with security levels following [1] and its extended and revisited version [2]. Sections IV and V describe calculus, types and security properties of the X$d\pi$-calculus with role-based access control following [3]. Section VI discusses related papers.

II. Modelling Dynamic Web Data

Our starting point is the X$d\pi$ of Gardner and Maffeis, introduced in [4], [5], which we equip with security levels. We model a peer-to-peer network as a set of connected locations, where each location has a security level and consists of a data tree and a process. Processes can, as in pure X$d\pi$, communicate with other processes, migrate to other locations and update the local data. The novelty is that all these actions are controlled by security levels.

The typed version of X$d\pi$ syntax is given in Table I. The presented data model is unordered edge-labelled rooted *tree* with leaves containing empty trees, scripts and pointers. A *script* is a static process embedded in a tree that can be activated by a process from the same location. A *path* identifies nodes in a tree. In a path, "$\mathtt{a}$" denotes a step along an edge $\mathtt{a}$, "$//$ " denotes any node, "$..$" a step back, "$\bullet$" the path from the root to the current node, "x" a variable and "$/$ " the path composition. A *pointer*, $p@\lambda$, refers to the set of data identified by the path p in the tree at the location λ. The symbol λ ranges over location names and variables. Other data terms, besides the scripts, the pointers and the trees, can be easily accomodated. There are three kinds of *processes*:

- the processes 0, $P|P$, $(\nu c^{T_v})P$, $\bar{\gamma}\langle v\rangle$ and $\gamma(x).P$ from π-calculus of Milner, Parrow and Walker, [6], [7], model local communication;
- the $\mathtt{go}$ command as in $d\pi$-calculus of Hennessy [8] allows processes to migrate from one location to another;
- the $\mathtt{run}$ command activates the execution of scripts that are embedded in the local data tree and the $\mathtt{update}$ command modifies the local data tree. These two commands allowing interaction between processes and data were introduced by Gardner and Maffeis in [4].

A *value* is either a channel name super-scripted with a value type, a script, a location name super-scripted with a security level, a path or a tree. A *pattern* is either a script pattern, or a pointer pattern, or a data-less tree variable, or a tree variable. The command $\mathtt{run}_p$ finds all the scripts in the local tree identified by the path p and it activates their parallel execution. The update command $\mathtt{update}_p(\chi, V)$ finds all the data terms identified by the path p and pattern matches these data terms with χ. For each successful pattern matching it replaces the found data with the term V in which the matched data substitutes χ. In order to improve readability of the examples we wrote $\mathtt{read}_p(\chi)$ instead of $\mathtt{update}_p(\chi, \chi)$ and $\mathtt{change}_p(\chi, V)$ instead of $\mathtt{update}_p(\chi, V)$ when $V \neq \chi$.

Reduction rules

The reduction relation $\rightarrow$ describes three forms of interactions:

- processes can communicate with each other within a location;
- processes can move between locations;
- processes can interact with the local data.

The reduction relation is the least relation on networks which is closed with respect to structural equivalence and reduction contexts. The definitions of structural equivalence, reduction

$T ::=$	tree	
$\emptyset$	empty rooted tree	
x	tree variable	
$T \mid T$	composition of trees, joining the roots	
$\mathtt{a}[T]$	edge labeled $\mathtt{a}$ with subtree T	
$\mathtt{a}[\Box\Pi]$	edge labeled $\mathtt{a}$ with script $\Box\Pi$	
$\mathtt{a}[p@\lambda]$	edge labeled $\mathtt{a}$ with pointer $p@\lambda$	
$p ::=$	$\mathtt{a} \mid // \mid .. \mid \bullet \mid x \mid p/p$	path
$V ::=$	$\Box\Pi \mid p@\lambda \mid T$	data term
$P ::=$	process	
0	the nil process	
$P \mid P$	parallel composition of processes	
$(\nu c^{Tv})P$	restriction of s channel name	
$\bar{\gamma}\langle v\rangle$	output of value v on channel γ	
$\gamma(x).P$	input parametrised by a variable x	
$!\gamma(x).P$	replication of an input process	
$\mathtt{go}\ \lambda.P$	migrates to location λ, continue as P	
$\mathtt{go}\ \circlearrowleft.P$	migrates to source location, continue as P	
$\mathtt{run}_p$	runs the scripts identified by path p	
$\mathtt{update}_p(\chi, V).P$	updates data identified by path p and matching with χ	
$v ::=$	$c^{Tv} \mid \Box P \mid l^h \mid p \mid T$	value
$\chi ::=$	$\Box x^j \mid y^\star@x^j \mid x^{DL} \mid x$	pattern
$\mathbf{N} ::=$	network	
$\mathbf{0}$	empty network	
$\mathbf{N} \mid \mathbf{N}$	parallel composition of networks	
$l^h[T \parallel P]$	location	
$(\nu c^{Tv})\mathbf{N}$	restriction of a channel name	

TABLE I
THE X$d\pi$ SYNTAX

$Ch(Tv)$	type of channels communicating values of type Tv
$Loc(i)$	type of locations at security level i
$Script(i)$	type of scripts at security level i
$Path$	type of paths, not containing $\bullet$
$PathLocal$	type of paths, possibly containing $\bullet$
$Pointer(i)$	type of pointers, not containing local paths, at security level i
$PointerLocal(i)$	type of pointers, possibly containing local paths, at security level i
$DTLTree$	type of data-less trees
$Tree$	type of trees, not containing local paths
$TreeLocal$	type of trees, possibly containing local paths
$Proc(i)$	type of processes, not containing local paths, at security level i
$ProcLocal(i)$	type of processes, possibly containing local paths, at security level i
Net	type of networks

where $i \in \mathcal{L}$ and Tv ranges over value types defined by

$Tv ::= Ch(Tv) \mid Loc(i) \mid Script(i) \mid Path^\star \mid DTLTree \mid Tree^\star$

TABLE II
THE SYNTAX OF X$d\pi$ TYPES

contexts and reduction relation can be found in [1] and [2]. Here we only give an example of reduction.

Example 1: Let the location of the online music store be named m, have a security level 2 and contain the data tree $T_\mathtt{s}$ and the process $P_\mathtt{o}$. If we take $T_\mathtt{s}$ to be the data tree (1) and $P_\mathtt{o}$ to be the process (2), then the network containing only location m reduces as follows

$$m^2[T_\mathtt{s} \parallel P_\mathtt{o}] \rightarrow m^2[T'_\mathtt{s} \parallel 0]$$

where $T'_\mathtt{s} \equiv \mathtt{store}\ [\mathtt{lyrics}\ [title^1] \mid [\mathtt{download}\ [demo^1]]$.

III. SECURITY LEVELS AND TYPES

A type system, in general, splits elements of the calculus into sets called types and makes certain behaviours (actions) illegal on the basis of the types that are thus assigned. The goal of our work is to verify the security properties by typing.

A. Types

Many type systems controlling the use of resources and the mobility of processes have been proposed for the $d\pi$ calculus [9] and for related calculi [10]–[12]. Our type system is based on types for locations, data and processes, expressing security levels and it is essentially inspired by the security types checking access rights for π-calculus of [13]. Its main goals are to control communication of values, access to data and migration of processes between locations. The syntax of types introduced in [2] is the content of Table II.

The access and mobility rights of a process depend on the security level of the "source" location of the process itself, i.e. of the location where the process was in the initial network or where the process was created by the activation of a script. More details on the types and the type assignment rules can be found in [1], [2].

B. Properties

Our system satisfies the property of subject reduction which means that a well-typed network reduces to a well-typed network. It is formally stated in the following theorem.

Theorem 1 (Subject reduction): Let $\vdash \mathbf{N} : Net$ and $\mathbf{N} \rightarrow \mathbf{N}'$, then $\vdash \mathbf{N}' : Net$.

Using the subject reduction, we can show some more meaningful properties of typed networks:

- PL0 A channel in a process whose source location has level h can communicate only the values whose security levels are less than or equal to h;
- PL1 A process whose source location has level h can migrate to a location of level j only if $j \leq h$;
- PL2 A process whose source location has level h can copy from the local tree only the data of level j with $j \leq h$;
- PL3 A process whose source location has level h can modify in the local tree only the data of level j with $j < h$, unless the process itself was generated by running a script of security level h in a tree at path p, and in this case it can modify scripts which are both of the security level h and reachable by the path p;
- PL4 A script of level j which is a leaf of a tree in a location of level i can be activated only if $j \leq i$.

We have formalised the network properties assured by our type system using the notions of network invariant and initial network as in [14]. For more details and all proofs we refer to the paper [2], while here we give an example.

Example 2: Let us consider the location m of the online music store of Example 1. We can assign type $Tree$ to the data tree $T_\mathtt{s}$. If the source location of the process $P_\mathtt{o}$ is at security level 3, then the location m has type Net. If the

source location of the process $P_{\mathtt{o}}$ is at security level 2, then the location m is not typeable in our typing system because, according to the security property PL4, a process whose source location is of level 2 can only modify data of security level 1. In other words, our typing system would rule out the user of security level 2, "pretending to be the owner", which aims to modify $song$ file.

IV. Role Based Access Control in X$d\pi$

In the example given in the Introduction we have stated the motivation for designing a model of dynamic web data in a RBAC scenario. In [3] we have equipped the X$d\pi$-calculus with roles and named the obtained calculus ®X$d\pi$-calculus: the syntax of ®X$d\pi$ is the content of Table III. As in the original calculus a *network* is a parallel composition of connected locations. Each *location* has a policy and consists of a data tree labelled with roles and a process with roles. We assume a countable set of roles $\mathcal{R}$, and use r, s, t to range over elements of $\mathcal{R}$. Let $(\mathcal{R}, \sqsubseteq)$ be a lattice and let $\bot, \top \in \mathcal{R}$ be its bottom and top element, respectively. The operation of join is denoted by $\sqcup$. By α, ρ, σ we denote non-empty sets of roles and by τ, ζ sets of roles containing the $\top$ element. The *trees* of ®X$d\pi$ are those of X$d\pi$ where each edge is assigned a set of roles containing the $\top$ element. *Pure processes* are essentially X$d\pi$ processes to which we add commands for administration of access rights. The commands `read` and `change` are in place of the `update` command of [4]. The new commands `enable` and `disable` change permissions to access data by adding or removing roles from edges in the local data tree. A *process with roles* is obtained from a pure process by assigning a set of roles ρ to it or as a parallel composition of such processes. Processes with (possibly different) roles can share private communication channels (restriction operator ν). Different processes can have different sets of roles and the same role can be assigned to different edges and different processes. In the syntax of *paths*, we do not consider paths containing // , .. and • and we decorate path edge labels with sets of roles. For simplicity, in the examples of this paper, we have omitted these decorations. There are other minor differences between ®X$d\pi$ and X$d\pi$ which we will omit.

$T ::=$	tree	
$\emptyset$	empty rooted tree	
x	tree variable	
$T \mid T$	composition of trees, joining the roots	
$\mathtt{a}^{\tau}[T]$	edge labeled $\mathtt{a}^{\tau}$ with subtree T	
$\mathtt{a}^{\tau}[\Box\Pi]$	edge labeled $\mathtt{a}^{\tau}$ with script $\Box\Pi$	
$\mathtt{a}^{\tau}[p@\lambda]$	edge labeled $\mathtt{a}^{\tau}$ with pointer $p@\lambda$	
$p ::= \mathtt{a}^{\alpha} \mid x \mid p/p$		path
$V ::= \Box\Pi \mid p@\lambda \mid T$		data term
$P ::=$	pure process	
0	the nil process	
$P \mid P$	parallel composition of processes	
$\bar{\gamma}\langle v\rangle$	output of value v on channel γ	
$\gamma(x).P$	input parametrised by a variable x	
$!\gamma(x).P$	replication of an input process	
$\mathtt{go}\ \lambda.R$	migrates to location λ, continue as P	
$\mathtt{run}_p$	runs the scripts identified by path p	
$\mathtt{read}_p(\chi).P$	reads data identified by path p and matching with χ	
$\mathtt{change}_p(\chi, V).P$	changes data identified by path p and matching with χ	
$\mathtt{enable}_p(r).P$	allows role r to access data identified by path p	
$\mathtt{disable}_p(r).P$	forbids role r to access data identified by path p	
$R ::=$	process with roles	
$P^{\neg\rho}$	single pure process with roles ρ assigned to it	
$R \mid R$	parallel composition of processes with roles	
$(\nu c^{Tv})R$	restriction of a channel name	
$v ::= c^{Tv} \mid \Box R \mid l \mid p \mid T$		value
$\chi ::= \Box x^{(\sigma,\mathcal{E},\mathcal{D})} \mid y^{(\alpha)}@x^{(\sigma,\mathcal{E},\mathcal{D})} \mid x^{(\sigma,\mathcal{E},\mathcal{D},\tau,\zeta)}$		pattern
$\mathbf{N} ::=$	network	
$\mathbf{0}$	empty network	
$\mathbf{N} \mid \mathbf{N}$	parallel composition of networks	
$l[T \parallel P]$	location	
$(\nu c^{Tv})\mathbf{N}$	restriction of a channel name	

TABLE III
The ®X$d\pi$ Syntax

Reduction rules

Processes with roles can, as in pure X$d\pi$, communicate with other processes, migrate to other locations and update the local data. All these actions are controlled by roles. Moreover, processes can administrate roles by enabling and disabling them. More details and the formal definition of reduction relation can be found in [3], while here we give an example.

Example 3: Let the location of the online music store be named m and contain the data tree $T_{\mathtt{s}}$ and the process $P_{\mathtt{o}}$. If we take $T_{\mathtt{s}}$ to be the data tree (3) and $P_{\mathtt{o}}$ to be the process $\mathtt{enable}_{\mathtt{store}}(\mathtt{visitor})^{\neg\{\mathtt{owner}\}}$, then the network containing only location m reduces as follows

$$m[T_{\mathtt{s}} \parallel P_{\mathtt{o}}] \rightarrow m[T'_{\mathtt{s}} \parallel 0]$$

where $T'_{\mathtt{s}}$ is

$$\mathtt{store}^{\{\mathtt{visitor}\}}[\mathtt{ly}.^{\{\mathtt{visitor}\}}[title] \mid [\mathtt{down}.^{\{\mathtt{user},\mathtt{visitor}\}}[song]].$$

V. Security Policies and Types

A location policy is the triple $(\sigma, \mathcal{E}, \mathcal{D})$, where σ is a set of roles, and $\mathcal{E}$ and $\mathcal{D}$ are subsets of $\{(\rho, r) : \rho \subseteq \mathcal{R},\ r \in \mathcal{R}\}$. The data accessibility policy is given by the set σ, the set of minimal roles a process is required to have to access the data at that location. The administration policy is given by the other sets which prescribe changes of data access rights as follows: if $(\rho, r) \in^{+} \mathcal{E}$, a process with roles ρ can give the permission to (enable) the role r to access the data; if $(\rho, r) \in^{-} \mathcal{D}$, a process with roles ρ can take the permission from (disable) the role r to access the data. $\in^{+}$ and $\in^{-}$ are defined in [3] as extensions of $\in$ in order to give more flexibility to the location policy.

A. Types

Given a location policy we can check if a data tree and a process conform to it. The syntax of ®X$d\pi$ types is the content of Table IV. Our type system assures that: if a process can access an edge in a well-typed tree, then the edge is connected to the root of the tree by a path whose edges are all accessible to that process; only processes agreeing with the location policy can be activated at a location and can migrate

$Ch(Tv)$	type of channels communicating values of type Tv
$Loc(\mathcal{P})$	type of locations with policy $\mathcal{P}$
$Script(\mathcal{P})$	type of scripts which can be activated at locations with policy $\mathcal{P}$
$Path(\alpha)$	type of paths having the last edge with set of roles α
$Pointer(\alpha)$	type of pointers whose path is typed by $Path(\alpha)$
$Tree(\mathcal{P},\tau,\zeta)$	type of trees, which can stay at locations with policy $\mathcal{P}$, with initial branches asking τ and which can be completely accessed by processes with at least one role of ζ
$Proc(\mathcal{P},\rho)$	type of pure processes, which can stay at locations with policy $\mathcal{P}$ and which can be assigned roles ρ
$ProcRole(\mathcal{P})$	type of processes with roles which can stay at locations with policy $\mathcal{P}$
Net	type of networks

Tv ranges over *value types* defined by:

$Tv ::= Ch(Tv) \mid Loc(\mathcal{P}) \mid Script(\mathcal{P}) \mid Path(\alpha) \mid Tree(\mathcal{P},\tau,\zeta)$

TABLE IV
THE SYNTAX OF ®X$d\pi$ TYPES

to it; a process can modify a subtree only if it can access all the edges of the subtree; agreeing with the location policy, a process can enable a role at an edge or disable a role from a subtree if it can access the path which identifies it.

Other common features of RBAC system we did not consider in [3], since we could smoothly add them to the present calculus, are: incompatible roles, static and dynamic separation of roles, limits on the number of users authorised for a given role.

B. Properties

Besides subject reduction property, we can prove the following relevant access control properties.

Properties of location policies and communication:

PR0 All trees and processes in a location agree with the location policy;

PR1 A process with roles can communicate only values with at least one characteristic role lower than or equal to one role of the process.

Properties of migration between locations:

PR2 A process with roles can migrate to another location only if it agrees with the policy of that location.

Properties of process access to local data trees:

PR3 A process with roles looks for a path in the local tree only if the path is accessible to the process.

PR4 A process with roles can get a data in the local tree only if the data is accessible to the process.

Properties of manipulation of local data trees by processes:

PR5 A script is activated in a location only if the corresponding process with roles agrees with the policy of that location;

PR6 A process with roles generated by a read command in a location agrees with the policy of that location;

PR7 A process with roles can erase a subtree of data only if it can access the whole subtree;

PR8 A tree built by a change command in a location agrees with the policy of that location;

PR9 A process with roles can add a role to an edge in the local tree only if this is allowed by the location policy;

PR10 A tree built by an enable command in a location agrees with the policy of that location;

PR11 A process with roles can erase a role from a subtree of the local tree only if this is allowed by the location policy;

PR12 A tree built by a disable command in a location agrees with the policy of that location.

More details on the location policies, the types, the type assignment rules and the security properties are discussed in [3] while here we give an example.

Example 4: Let us consider the location m of the online music store with location policy (4) containing the data tree $T_{\mathtt{s}}$ of Example 3 and the process $P_{\mathtt{u}}$. The process $P_{\mathtt{u}}$ cannot be $\mathtt{enable}_{\mathtt{st}}(\mathtt{visitor})^{\neg\{\mathtt{user}\}}$, i.e. a process with role `user` aiming to give permission to the role `visitor` to access the $file$. Since $(\{\mathtt{user}\},\mathtt{visitor}) \notin \{(\{\mathtt{owner}\},\mathtt{visitor})\}$, the process $P_{\mathtt{u}}$ does not agree with the store's policy and according to the security property PR0, the location m is not typeable in our typing system. In other words, our typing system would rule out the user, "pretending to be the owner", wishing to enable the visitors to access the $song$.

VI. RELATED WORK

The X$d\pi$ calculus [4], [5] models both localised, mobile processes and distributed, dynamic, semi-structured data, allowing to represent data-sharing applications. It can be seen as an extension of the Active XML model [15]. The locations and the processes of X$d\pi$ are essentially those of $d\pi$ [8], [9] enriched with capabilities for data manipulation. The only difference is that a process in $d\pi$ can migrate to a location independently from the existence of the location itself in the current network, while in X$d\pi$ such an existence is a necessary condition for migration. The data trees of X$d\pi$ are related to those in [16], [17] and the treatment of shared distributed data is inspired by [18].

In the SafeDpi calculus [19] parameterised code may be sent between locations and types restrict the capabilities and access rights of any processes launched by incoming code. Co-actions have been introduced for ambient calculi as a basic mechanism for regulating access to locations and use of their resources [11], [20], [21]. More refined controls for ambient calculi include passwords [22], [23], classifications in groups [12], [24], mandatory access control policies [25], membranes regulating the interaction between computing bodies and external environments [26].

Role based access control has been introduced in the seventies and first formalised by Ferraiolo and Kuhn [27]. There is a large amount of literature on models and implementations for RBAC, we only mention [28]–[31]. The standard defined in 2004 is currently under revision by the Committe CS1.1 within the International Committee for Information Technologies Standards [32].

The most related papers to [3] are [33] and [34]. Braghin et al. [33] equip the π-calculus with the notion of user: they tag

processes with names of users and with sets of roles. Processes can activate and deactivate roles. A mapping between roles and users and a mapping between read/write actions and roles control access rights. A type discipline statically guaranties that systems not respecting the above mappings are rejected. Compagnoni et al. [34] define a boxed ambient calculus extended with a distributed RBAC mechanism where each ambient controls its own access policy. A process is associated with an owner and a set of activated roles that grant permissions for mobility and communication. The calculus includes primitives to activate and deactivate roles. The behaviour of these primitives is determined by the process's owner, its current location and its currently activated roles.

REFERENCES

[1] M. Dezani-Ciancaglini, S. Ghilezan, and J. Pantovic, "Security types for dynamic web data," in *TGC'06*, ser. LNCS, vol. 4661. Springer, 2007, pp. 263–280.

[2] M. Dezani-Ciancaglini, S. Ghilezan, J. Pantovic, and D. Varacca, "Security types for dynamic web data," *Theoretical Computer Science*, vol. 402, no. 2-3, pp. 156–171, 2008.

[3] M. Dezani-Ciancaglini, S. Ghilezan, S. Jakšić, and J. Pantović, "Types for role-based access control of dynamicweb data," in *WFLP'10*, ser. LNCS, 2011, to appear.

[4] P. Gardner and S. Maffeis, "Modelling dynamic web data," *Theoretical Computer Science*, vol. 342, no. 1, pp. 104–131, 2005.

[5] S. Maffeis and P. Gardner, "Behavioural equivalencies for dynamic web data," in *TCS'04*. Kluwer, 2004, pp. 541–554.

[6] R. Milner, J. Parrow, and D. Walker, "A calculus of mobile processes, I, II," *Information and Computation*, vol. 100, no. 1, pp. 1–40,41–77, 1992.

[7] D. Sangiorgi and D. Walker, *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

[8] M. Hennessy, *A Distributed Pi-calculus*. Cambridge University Press, 2007.

[9] M. Hennessy and J. Riely, "Resource access control in systems of mobile agents," *Information and Computation*, vol. 173, no. 1, pp. 82–120, 2002.

[10] R. D. Nicola, G. Ferrari, R. Pugliese, and B. Venneri, "Types for access control," *Theoretical Computer Science*, vol. 240, no. 1, pp. 215–254, 2000.

[11] G. Castagna, J. Vitek, and F. Z. Nardelli, "The Seal calculus," *Information and Computation*, vol. 201, no. 1, pp. 1–54, 2005.

[12] L. Cardelli, G. Ghelli, and A. D. Gordon, "Types for the ambient calculus," *Information and Computation*, vol. 177, no. 2, pp. 160–194, 2002.

[13] M. Hennessy and J. Riely, "Information flow vs resource access in the asynchronous π-calculus," *ACM Transactions on Programming Languages and Systems*, vol. 5, pp. 566–591, 2003.

[14] A. Ahern and N. Yoshida, "Formalising Java RMI with explicit code mobility," in *OOPSLA'05*. ACM Press, 2005, pp. 403–422.

[15] S. Abiteboul, O. Benjelloun, B. Cautis, and T. Milo, "Active XML, security and access control," in *SBBD'04*, 2004, pp. 13–22.

[16] S. Abiteboul, P. Buneman, and D. Suciu, *Data on the Web: From Relations to Semistructured Data and XML*, ser. Data Management Systems. Morgan Kaufmann, 1999.

[17] L. Cardelli and G. Ghelli, "A Query Language Based on the Ambient Logic," in *ESOP'01*, ser. LNCS, vol. 2028. Springer, 2004, pp. 1–22, invited Paper.

[18] A. Sahuguet, "ubQL: A distributed query language to program distributed query systems," Ph.D. dissertation, Penn University, 2002.

[19] M. Hennessy, J. Rathke, and N. Yoshida, "SafeDpi: A language for controlling mobile code," *Acta Informatica*, vol. 42, no. 4-5, pp. 227–290, 2005.

[20] F. Levi and D. Sangiorgi, "Controlling interference in ambients," *Transactions on Programming Languages and Systems*, vol. 25, no. 1, pp. 1–69, 2003.

[21] P. Garralda, E. Bonelli, A. Compagnoni, and M. Dezani-Ciancaglini, "Boxed ambients with communication interfaces," *Mathematical Structures in Computer Science*, vol. 17, pp. 1–59, 2007.

[22] M. Merro and M. Hennessy, "A bisimulation-based semantic theory of safe ambients," *ACM Transactions on Programming Languages and Systems*, vol. 28, no. 2, pp. 290–330, 2006.

[23] M. Bugliesi, S. Crafa, M. Merro, and V. Sassone, "Communication and mobility control in boxed ambients," *Information and Computation*, vol. 202, no. 1, pp. 39–86, 2005.

[24] M. Coppo, M. Dezani-Ciancaglini, and E. Giovannetti, "Types for ambient and process mobility," *Mathematical Structures in Computer Science*, vol. 18, pp. 221–290, 2008.

[25] M. Bugliesi, G. Castagna, and S. Crafa, "Access control for mobile agents: The calculus of boxed ambients," *ACM Transactions on Programming Languages and Systems*, vol. 26, no. 1, pp. 57–124, 2004.

[26] D. Gorla, M. Hennessy, and V. Sassone, "Security policies as membranes in systems for global computing," *Logical Methods in Computer Science*, vol. 1, no. 3:2, p. 331353, 2005.

[27] D. F. Ferraiolo, D. R. Kuhn, and R. S. Sandhu, "Rôle-based access control," in *NIST-NSA National Computer Security Conference*, 1992, pp. 554–563.

[28] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[29] D. F. Ferraiolo, J. F. Barkley, and D. R. Kuhn, "A role-based access control model and reference implementation within a corporate intranet," *ACM Transactions on Information and System Security*, vol. 2, no. 1, pp. 34–64, 1999.

[30] S. Osborn, R. S. Sandhu, and Q. Munawer, "Configuring role-based access control to enforce mandatory and discretionary access control policies," *ACM Transactions on Information and System Security*, vol. 3, no. 2, pp. 85–106, 2000.

[31] D. F. Ferraiolo, R. S. Sandhu, S. I. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST standard for role-based access control," *ACM Transactions on Information and System Security*, vol. 4, no. 3, pp. 224–274, 2001.

[32] D. R. Kuhn, E. J. Coyne, and T. R. Weil, "Adding attributes to role-based access control," *Computer*, vol. 43, no. 6, pp. 79–81, 2010.

[33] C. Braghin, D. Gorla, and V. Sassone, "Role-based access control for a distributed calculus," *Journal of Computer Security*, vol. 14, no. 2, pp. 113–155, 2006.

[34] A. B. Compagnoni, E. L. Gunter, and P. Bidinger, "Role-based access control for boxed ambients," *Theoretical Computer Science*, vol. 398, no. 1-3, pp. 203–216, 2008.

Silvia Ghilezan received the PhD degree in mathematics in 1993 from the University of Novi Sad, Novi Sad, Serbia. She joined the Faculty of Technical Sciences, University of Novi Sad, Serbia in 1984 and currently holds a Professor position. On several occasions she has held visiting positions at McGill University, Canada, École Normale Supérieure de Lyon, France, University of Turin, Italy and Catholic University, The Netherlands. Her research interests are in the areas of logic in computer science, foundations of computer science, type theory, theory of mobile processes and mathematical linguistics.

Svetlana Jakšić received the M.Sc. degree in mathematics from the University of Novi Sad, Serbia, in 2008. She is currently a Ph.D. student and a teaching assistant at the Faculty of Technical Sciences, University of Novi Sad. Her research interests are in the areas of process calculi, concurrency theory and type theory.

Jovanka Pantović received the Ph.D. degree in mathematics from the University of Novi Sad, Serbia, in 2000. She joined the Faculty of Technical Sciences, University of Novi Sad, in 1993, where she is currently a Professor. Her research interests are in the areas of multiple-valued logic, universal algebra, discrete mathematics, and theory of mobile processes.

Mariangiola Dezani-Ciancaglini is full professor of "Foundations of Computer Science" at the University of Torino since 1981. She has co-authored more than 100 papers published in prestigious scientific journals and in proceedings of international conferences. In the '80s she introduced with other researchers the intersection type assignment systems, which were largely used as finitary descriptions of lambda-models. More recently, she studied type systems for object-oriented and ambient calculi and session types for assuring safety of communication protocols.